```
In [1]: import pandas
```

```
In [2]: import pandas as pd
```

```
In [3]: #change default value for displaying rows and column
```

```
In [4]: #display default maximum rows
        pd.get_option("display.max_rows")

Out[4]: 60
```

```
In [5]: #display default maximum column
        pd.get_option("display.max_columns")

Out[5]: 20
```

```
In [6]: #set maximum rows
        pd.set_option("display.max_rows",80)
```

```
In [7]: #set maximum column
        pd.set_option("display.max_columns",42)
```

```
In [8]: #reset to default value

        #pd.reset_option("display.max_rows")
        #pd.reset_option("display.max_colums")
```

Reading dataset using pandas

flat files- read.csv(),to_csv()

Excel files- read_excel(),ExcelWriter(),to_excel()

JSON files - read_json(),to_json()

HTML tables- read_html(),to_html()

SAS files - read_sas()

STATA files- read_stata(),to_stata()

SQL files - read_sql(),read_sql_query(),read_sql_table(),tc

```
In [9]: import pandas as pd
```

```
In [10]: P=pd.read_csv("c:/Users/qumrul hoda/Downloads/blackfriday - blackfriday.csv")
```

```
In [11]: P
```

Out[11]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 12 columns

```
In [12]: type(P)
```

```
Out[12]: pandas.core.frame.DataFrame
```

```
In [13]: #find shape of dataframes i.e no. of rows and columns
```

```
In [14]: P.shape
```

```
Out[14]: (550068, 12)
```

```
In [15]: #dimension of dataframe
         P.ndim
```

```
Out[15]: 2
```

```
In [16]: #print top 10 rows and last 10 rows from dataset
```

```
In [17]: P.head(10)
```

Out[17]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	

In [18]:
```
P.tail(10)
```

Out[18]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
550058	1006024	P00372445	M	26-35	12	A	0
550059	1006025	P00370853	F	26-35	1	B	1
550060	1006026	P00371644	M	36-45	6	C	1
550061	1006029	P00372445	F	26-35	1	C	1
550062	1006032	P00372445	M	46-50	7	A	3
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

```
In [19]: #checking for duplicate rows
         P.duplicated()
```

```
Out[19]: 0         False
         1         False
         2         False
         3         False
         4         False
                   ...
         550063    False
         550064    False
         550065    False
         550066    False
         550067    False
         Length: 550068, dtype: bool
```

```
In [20]: sum(P.duplicated())
```

```
Out[20]: 0
```

```
In [21]: #create new dataset with all product related column
         #lets check columns present in dataframe
         P.columns
```

```
Out[21]: Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
                'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
                'Product_Category_2', 'Product_Category_3', 'Purchase'],
               dtype='object')
```

```
In [22]: [i for i in P.columns if "Product" in i]
```

```
Out[22]: ['Product_ID',
          'Product_Category_1',
          'Product_Category_2',
          'Product_Category_3']
```

```
In [23]: #lets creat new dataframe with the product related column
         P_Product=P[["Product_ID","Product_Category_1","Product_Category_2","Product_Catego
```

```
In [24]: P_Product
```

Out[24]:

	Product_ID	Product_Category_1	Product_Category_2	Product_Category_3
0	P00069042	3	NaN	NaN
1	P00248942	1	6.0	14.0
2	P00087842	12	NaN	NaN
3	P00085442	12	14.0	NaN
4	P00285442	8	NaN	NaN
...	...	...	...	...
550063	P00372445	20	NaN	NaN
550064	P00375436	20	NaN	NaN
550065	P00375436	20	NaN	NaN
550066	P00375436	20	NaN	NaN
550067	P00371644	20	NaN	NaN

550068 rows × 4 columns

In [25]:
```
P.axes
```

Out[25]:
```
[RangeIndex(start=0, stop=550068, step=1),
 Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
        'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
        'Product_Category_2', 'Product_Category_3', 'Purchase'],
       dtype='object')]
```

In [26]:
```
#value returns the series of values as ndarray
P.Gender.values
```

Out[26]:
```
array(['F', 'F', 'F', ..., 'F', 'F', 'F'], dtype=object)
```

In [27]:
```
P.size
```

Out[27]:
```
6600816
```

In [28]:
```
P.Gender.size
```

Out[28]:
```
550068
```

In [29]:
```
#check for all columns datatype and related counts
#lets use info() function
P.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [30]:
```
#lets check only datatype of each columns
P.dtypes
```

Out[30]:
```
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                      int64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                  int64
Product_Category_1              int64
Product_Category_2            float64
Product_Category_3            float64
Purchase                        int64
dtype: object
```

In [31]:
```
#Change datatype of "Purchase" to float
P["Purchase"]=P["Purchase"].astype("Float64")
```

In [32]:
```
P.dtypes
```

Out[32]:
```
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                      int64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                  int64
Product_Category_1              int64
Product_Category_2            float64
Product_Category_3            float64
Purchase                      Float64
dtype: object
```

In [33]:
```
P.head()
```

Out[33]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [34]:
```
#change it back to int
P["Purchase"]=P["Purchase"].astype("int64")
```

In [35]:
```
P.dtypes
```

Out[35]:
```
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                      int64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                  int64
Product_Category_1              int64
Product_Category_2            float64
Product_Category_3            float64
Purchase                        int64
dtype: object
```

In [36]:
```
#Generate descriptive statistical values for numerical columns
P.describe()
```

Out[36]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Proc
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	

In [37]:
```
#Try to use same funtion on categorical columns.
#apart from int and float ,we have datatype as object.
P.describe(include=["object"])
```

Out[37]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

In [38]:
```
#get percentage distribution of each product id avaailable in dataset and find wit
#we can use value_counts
```

In [39]:
```
#lets check for Product_ID
P["Product_ID"].value_counts()
```

Out[39]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ... 
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

In [40]:
```
#Get normalized value
P["Product_ID"].value_counts(normalize=True)
```

Out[40]:
```
P00265242    0.003418
P00025442    0.002936
P00110742    0.002931
P00112142    0.002840
P00057642    0.002672
               ...   
P00314842    0.000002
P00298842    0.000002
P00231642    0.000002
P00204442    0.000002
P00066342    0.000002
Name: Product_ID, Length: 3631, dtype: float64
```

In [41]:
```
#multiply by 100 to get percentage value and round it up to 3 decimal place
round(P["Product_ID"].value_counts(normalize=True)*100,3)
```

Out[41]:
```
P00265242    0.342
P00025442    0.294
P00110742    0.293
P00112142    0.284
P00057642    0.267
             ...  
P00314842    0.000
P00298842    0.000
P00231642    0.000
P00204442    0.000
P00066342    0.000
Name: Product_ID, Length: 3631, dtype: float64
```

In [42]:
```
#check sum of all percentage value
round(P["Product_ID"].value_counts(normalize=True)*100,3).sum()
```

Out[42]:
```
99.97600000000001
```

Handling missing Values

check for columns having null values and count of null containing rows.

pandas provides isna() and notna() functions to detect "NA" Values. detect "NA" values in the dataframe:df.isna().sum() detect "NA" values in aparticular column in the dataframe: pd.isna(df["col_name"]) ,df["col_name"].notna() isnull() is just an alias of the isna() method in pandas source code.

In [43]:
```
#lets check on our dataset
P.isnull().sum()
```

Out[43]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

We can see 2 column i.e Product_category_2 and Product_category_3 with missing values and count is 173638 and 383247 respectively.

In [44]:
```
#print Product_category_2
P.Product_Category_2
```

Out[44]:
```
0          NaN
1          6.0
2          NaN
3         14.0
4          NaN
          ...
550063     NaN
550064     NaN
550065     NaN
550066     NaN
550067     NaN
Name: Product_Category_2, Length: 550068, dtype: float64
```

In [45]:
```
#print Product_category_3
P.Product_Category_3
```

Out[45]:

```
0            NaN
1           14.0
2            NaN
3            NaN
4            NaN
            ...
550063       NaN
550064       NaN
550065       NaN
550066       NaN
550067       NaN
Name: Product_Category_3, Length: 550068, dtype: float64
```

Sometimes missing values are encoded in different ways.They can appear as NaN,NA,?,zeros,xx,or a blank space. But Pandas always recognise missing values as NaN.So it is essential that we should first convert all the ?,zero,xx, to NaN.if the missing values is not identified as NaN.then we have to first convert or replace such non NaN entry with a Nan.

Convert "?" to NaN df[df=="?"]=np.nan

In [46]:

```
#Handle missing value using dropping and imputing both options.
```

In [47]:

```
#lets first try with drop option
P2=P.drop("Product_Category_2",axis=1,inplace=False)
```

In [48]:

```
P2
```

Out[48]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 11 columns

In [49]:
```
P2.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_3          166821 non-null  float64
 10  Purchase                    550068 non-null  int64  
dtypes: float64(1), int64(5), object(5)
memory usage: 46.2+ MB
```

In [50]:
```
#verify with isnull()
P2.isnull().sum()
```
Out[50]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_3            383247
Purchase                           0
dtype: int64
```

In [51]:
```
#get index for all rows with Product_Category_3 missing
P2.Product_Category_3.isnull()
```
Out[51]:
```
0          True
1         False
2          True
3          True
4          True
          ...  
550063     True
550064     True
550065     True
550066     True
550067     True
Name: Product_Category_3, Length: 550068, dtype: bool
```

In [52]:
```
P2[P2.Product_Category_3.isnull()].index
```
Out[52]:
```
Int64Index([     0,      2,      3,      4,      5,      7,      8,      9,
                10,     11,
            ...
            550058, 550059, 550060, 550061, 550062, 550063, 550064, 550065,
            550066, 550067],
           dtype='int64', length=383247)
```

In [53]:
```
#drop Product_Category_3 using axis 0
P2.drop(P2[P2.Product_Category_3.isnull()].index,axis=0,inplace=False)
```

Out[53]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
1	1000001	P00248942	F	0-17	10	A	2
6	1000004	P00184942	M	46-50	7	B	2
13	1000005	P00145042	M	26-35	20	A	1
14	1000006	P00231342	F	51-55	9	A	1
16	1000006	P0096642	F	51-55	9	A	1
...	...	...	...	...	...	...	...
545902	1006039	P00064042	F	46-50	0	B	4+
545904	1006040	P00081142	M	26-35	6	B	2
545907	1006040	P00277642	M	26-35	6	B	2
545908	1006040	P00127642	M	26-35	6	B	2
545914	1006040	P00217442	M	26-35	6	B	2

166821 rows × 11 columns

In [54]:
```
#verify using info()
P2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_3          166821 non-null  float64
 10  Purchase                    550068 non-null  int64  
dtypes: float64(1), int64(5), object(5)
memory usage: 46.2+ MB
```

In [55]:
```
#input using forward filling
P=P.fillna(method="pad")
```

In [56]:
```
P
```

Out[56]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 12 columns

In [57]:
```
#verify using isnull()
P.isnull().sum()
```

Out[57]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            1
Product_Category_3            1
Purchase                      0
dtype: int64
```

We can see that the Product_Category_2 and Product_Category_3 have 1 missing value.We can use the head() to check this.

In [58]:
```
#print both columns
P[["Product_Category_2","Product_Category_3"]].head()
```

Out[58]:

	Product_Category_2	Product_Category_3
0	NaN	NaN
1	6.0	14.0
2	6.0	14.0
3	14.0	14.0
4	14.0	14.0

In [59]:
```
#input using backward filling
P=P.fillna(method="backfill")
```

In [60]:
```
#verify using isnull()
P.isnull().sum()
```

Out[60]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

indexing and slicing in pandas

TASK-Print age and occupation column using loc and select 1st,5th and 10th rows with 1st,4th and 7th columns using iloc. .loc-label based .iloc-integer based

In [61]:
```
#make a copy of dataframe
df=P.copy()
```

In [62]:
```
df
```

Out[62]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 12 columns

In [63]:
```
#select first row of dataframe using loc
df.loc[0,"Product_ID"]
```

Out[63]:
```
'P00069042'
```

In [64]:
```
#print purchase for all rows using loc
df.loc[:,"Purchase"]
```

Out[64]:
```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550063     368
550064     371
550065     137
550066     365
550067     490
Name: Purchase, Length: 550068, dtype: int64
```

In [65]:
```
#select first five rows for specific column purchase
df.loc[:5,"Purchase"]
```

Out[65]:
```
0     8370
1    15200
2     1422
3     1057
4     7969
5    15227
Name: Purchase, dtype: int64
```

In [66]:
```
#print age and occupation using loc
df.loc[0:4,["Age","Occupation"]]
```

Out[66]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16

In [67]:
```
#lets try to print all columns for 2nd,3rd and 4th rows
df.loc[2:4]
```

Out[67]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

Integer position based indexing using .iloc indexer

In [68]:
```
#print first row using iloc
df.iloc[0]
```

Out[68]:
```
User_ID                         1000001
Product_ID                    P00069042
Gender                                F
Age                                0-17
Occupation                           10
City_Category                         A
Stay_In_Current_City_Years            2
Marital_Status                        0
Product_Category_1                    3
Product_Category_2                  6.0
Product_Category_3                 14.0
Purchase                           8370
Name: 0, dtype: object
```

In [69]:
```
#select last row of dataframe using iloc
df.iloc[-1]
```

Out[69]:
```
User_ID                          1006039
Product_ID                     P00371644
Gender                                 F
Age                                46-50
Occupation                             0
City_Category                          B
Stay_In_Current_City_Years            4+
Marital_Status                         1
Product_Category_1                    20
Product_Category_2                   2.0
Product_Category_3                  11.0
Purchase                             490
Name: 550067, dtype: object
```

In [70]:
```
#select first five columns of dataframe with all rows using iloc
df.iloc[:,0:5]
```

Out[70]:

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 5 columns

In [71]:
```
#select 1st ,5thand 10th rows with 1st,4th and 7th columns using iloc
df.iloc[[0,4,9],[0,3,6]]
```

Out[71]:

	User_ID	Age	Stay_In_Current_City_Years
0	1000001	0-17	2
4	1000002	55+	4+
9	1000005	26-35	1

Task- fetch row having maximum purchase amount with complete row details Pandas provide two function idxmax() and idxmin() that return index of first occurrence of maximum or minimum values over requested axix. NA/null values are excluded from the output.

In [72]:
```
#get index of first occurrence of maximum purchase value
df["Purchase"].idxmax()
```

Out[72]:
```
87440
```

```
In [73]: #get values of maximum purchase amount
         df.Purchase[df["Purchase"].idxmax()]
```

```
Out[73]: 23961
```

```
In [74]: #get the row with the maximum purchase value
         df[df["Purchase"]==23961]
```

Out[74]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1
370891	1003160	P00052842	M	26-35	17	C	3

```
In [75]: #get the row with maximum purchase value using loc
         df.loc[df[df["Purchase"]==23961].index]
```

Out[75]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1
370891	1003160	P00052842	M	26-35	17	C	3

so there is three users with maximum amount of purchase 23961

Task- get the purchase amount from 3rd rows Pandas also provide at() and iat() function to access a single value for a row and column pair by lable or by integer position.

```
In [76]: #get value at 3rd row and purchase column pair
         df.at[2,"Purchase"]
```

```
Out[76]: 1422
```

```
In [77]: df.head()
```

Out[77]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [78]:
```
#get value at 3rd row and 11th column pair
df.iat[2,11]
```

Out[78]:
```
1422
```

Task- find the purchase amount for a user_id(1006039) and product_id(P00371644) We can also use Boolean to filter and select the data | for or & for and ~ for not

In [79]:
```
#get the purchase amount with a given user_id and product_id
df.loc[((df["User_ID"]==1006039) & (df["Product_ID"]=="P00371644")),"Purchase"]
```

Out[79]:
```
550067    490
Name: Purchase, dtype: int64
```

In [80]:
```
#task- Find the user those are in city "A" with more than 4 years and purchase amou
#get the purchase amount with a given user_id and product_id
df[(df["City_Category"]=="A") & (df["Stay_In_Current_City_Years"]=="4+")&(df["Purch
```

Out[80]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
98	1000022	P00351142	M	18-25	15	A	4+
100	1000022	P00195942	M	18-25	15	A	4+
102	1000022	P0098242	M	18-25	15	A	4+
103	1000022	P00262242	M	18-25	15	A	4+
416	1000073	P00351142	M	18-25	4	A	4+
...	...	...	...	...	...	...	...
545791	1006019	P00279442	M	26-35	0	A	4+
545792	1006019	P00262342	M	26-35	0	A	4+
545793	1006019	P00028842	M	26-35	0	A	4+
545794	1006019	P00070342	M	26-35	0	A	4+
545832	1006028	P0097242	M	18-25	4	A	4+

6947 rows × 12 columns

Task-Discard all females users those are in city "B" with 3 years and purchase amount less than 5000.

In [81]:
```
#get the purchase amount with a given user_id and product_id
df[~((df["Gender"]=="F")&(df["City_Category"]=="B")&(df["Stay_In_Current_City_Years
```

Out[81]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

548117 rows × 12 columns

Task- Find the record in dataset with below details.
[1006038,"P00375436","F","55+",1,"C","2",0,20,2,0,11,0,365]

DataFrame also has an isin() method.when calling isin,we pass a set of values as either an array or dict.If values is an array,isin returns a Dataframe of booleans that is the same shape as the original DataFrame,with True wherever the element is in the sequence values.

In [82]:
```
#lets use isin funtion for searching row with given values.
values=[1006038,"P00375436","F","55+",1,"C","2",0,20,2,0,11,0,365]
df_indexed=df.isin(values)
```

In [83]:
```
df_indexed
```

Out[83]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	False	False	True	False	False	False	True
1	False	False	True	False	False	False	True
2	False	False	True	False	False	False	True
3	False	False	True	False	False	False	True
4	False	False	False	True	False	True	False
...	...	...	...	...	...	...	...
550063	False	False	False	False	False	False	False
550064	False	True	True	False	True	True	False
550065	False	True	True	False	False	False	False
550066	True	True	True	True	True	True	True
550067	False	False	True	False	True	False	False

550068 rows × 12 columns

In [84]:
```
#lets use all condition
#we can combine DataFrame isin with the any() and all methods to quickly select sul
df_indexed=df.isin(values).all(axis=1)
```

In [85]:
```
df[df_indexed]
```

Out[85]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
550066	1006038	P00375436	F	55+	1	C	2

Task-Visualize records with occcupation value 10 and mask everything left.

In [86]:
```
#lets use mask function to get only rows with occupation 10.
newdf=df.mask(df["Occupation"]!=10)
newdf.head(10)
```

Out[86]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
0	1000001.0	P00069042	F	0-17	10.0	A	2	
1	1000001.0	P00248942	F	0-17	10.0	A	2	
2	1000001.0	P00087842	F	0-17	10.0	A	2	
3	1000001.0	P00085442	F	0-17	10.0	A	2	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Sorting in pandas

-sorting by label -sorting by actual value

Task-sort dataset row wise and column wise

Sorting By label

we can use the sort_index() method to sort the object by labels.

In [87]:
```
#sort dtaset row wise
df.sort_index()
```

Out[87]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 12 columns

In [88]:

```
#sort dataset column wiaw
df.sort_index(axis=1)
```

Out[88]:

	Age	City_Category	Gender	Marital_Status	Occupation	Product_Category_1	Product_Cat
0	0-17	A	F	0	10	3	
1	0-17	A	F	0	10	1	
2	0-17	A	F	0	10	12	
3	0-17	A	F	0	10	12	
4	55+	C	M	0	16	8	
...	...	...	...	...	...	...	
550063	51-55	B	M	1	13	20	
550064	26-35	C	F	0	1	20	
550065	26-35	B	F	1	15	20	
550066	55+	C	F	0	1	20	
550067	46-50	B	F	1	0	20	

550068 rows × 12 columns

In [89]:
```
#sort row wise in descending order
df.sort_index(ascending=False)
```

Out[89]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
550067	1006039	P00371644	F	46-50	0	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550065	1006036	P00375436	F	26-35	15	B	4+
550064	1006035	P00375436	F	26-35	1	C	3
550063	1006033	P00372445	M	51-55	13	B	1
...	...	...	...	...	...	...	...
4	1000002	P00285442	M	55+	16	C	4+
3	1000001	P00085442	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
0	1000001	P00069042	F	0-17	10	A	2

550068 rows × 12 columns

Sorting By values

Task-Find top 20 most revenue generated customer and their purchase product id

Pandas provides sort_values() method to sort by values.it accepts a by argument which will use the column name of the DataFrame with which the values are to be sorted.

In [90]:

```
#lets sort dataset using purchase column
df.sort_values(by=["Purchase"])
```

Out[90]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
549221	1004806	P00370293	M	26-35	17	C	2
549477	1005184	P00370293	M	18-25	20	B	4+
547819	1002802	P00370853	M	36-45	20	B	2
548027	1003105	P00370853	M	36-45	12	C	4+
547538	1002402	P00370853	M	46-50	17	B	4+
...	...	...	...	...	...	...	...
292083	1003045	P00052842	M	46-50	1	B	2
503697	1005596	P00117642	M	36-45	12	B	1
370891	1003160	P00052842	M	26-35	17	C	3
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1

550068 rows × 12 columns

In [91]:

```
#sort by multiple column
df.sort_values(by=["Age","Purchase"]).head(10)
```

Out[91]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
546045	1000194	P00370853	F	0-17	10	C	3
546449	1000775	P00370853	M	0-17	17	C	1
550024	1005973	P00370293	M	0-17	10	C	4+
545971	1000086	P00370853	F	0-17	10	C	3
549145	1004707	P00370293	M	0-17	0	C	4+
549275	1004883	P00370293	F	0-17	10	C	1
546877	1001421	P00370293	F	0-17	10	A	1
548545	1003865	P00370853	F	0-17	10	C	2
546531	1000888	P00370853	F	0-17	10	C	1
546779	1001280	P00370853	M	0-17	10	C	1

In [92]:
```
#sort in decending order
df.sort_values(by="Purchase",ascending=False)
```

Out[92]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
370891	1003160	P00052842	M	26-35	17	C	3
93016	1002272	P00052842	M	26-35	0	C	1
87440	1001474	P00052842	M	26-35	4	A	2
503697	1005596	P00117642	M	36-45	12	B	1
321782	1001577	P00052842	M	55+	0	C	1
...	...	...	...	...	...	...	...
546379	1000671	P00370853	M	18-25	4	C	0
546185	1000391	P00370293	M	46-50	11	C	2
547032	1001649	P00370293	M	18-25	19	C	2
546181	1000387	P00370293	F	36-45	7	C	0
549221	1004806	P00370293	M	26-35	17	C	2

550068 rows × 12 columns

In [93]:
```
#lets find top 20 using iloc
top20=df.sort_values(by=["Purchase"],ascending=False).iloc[:20,:]
```

In [94]:
```
top20
```

Out[94]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
370891	1003160	P00052842	M	26-35	17	C	3
93016	1002272	P00052842	M	26-35	0	C	1
87440	1001474	P00052842	M	26-35	4	A	2
503697	1005596	P00117642	M	36-45	12	B	1
321782	1001577	P00052842	M	55+	0	C	1
349658	1005848	P00119342	M	51-55	20	A	0
292083	1003045	P00052842	M	46-50	1	B	2
298378	1003947	P00116142	M	26-35	0	C	3
437804	1001387	P00086242	F	51-55	13	B	1
229329	1005367	P00085342	M	18-25	4	A	1
416883	1004117	P00161842	M	18-25	4	B	4+
7542	1001178	P00116142	M	55+	0	C	1
373300	1003511	P00085342	M	51-55	0	C	2
33268	1005102	P00052842	M	26-35	12	C	2
388010	1005716	P00052842	M	0-17	10	C	4+
449656	1003301	P00086242	F	26-35	2	B	3
366333	1002359	P00085342	M	55+	13	C	1
54364	1002274	P00052842	M	18-25	2	B	3
56879	1002788	P00085342	M	55+	1	B	0
68926	1004520	P00116142	M	26-35	4	C	1

In [95]:
```
#get list of top 20 user id
top20.User_ID.values
```

Out[95]:
```
array([1003160, 1002272, 1001474, 1005596, 1001577, 1005848, 1003045,
       1003947, 1001387, 1005367, 1004117, 1001178, 1003511, 1005102,
       1005716, 1003301, 1002359, 1002274, 1002788, 1004520], dtype=int64)
```

In [96]:
```
#visualize products include in top 20
top20.Product_ID.value_counts()
```

```
Out[96]: P00052842    8
         P00085342    4
         P00116142    3
         P00086242    2
         P00117642    1
         P00119342    1
         P00161842    1
         Name: Product_ID, dtype: int64
```

Exploring categorical data

Task-Find which age group is much active for purchasing product from website

```
In [97]: #lets use unique to get distinct values
         df["Gender"].unique()
```

```
Out[97]: array(['F', 'M'], dtype=object)
```

```
In [98]: #lets use value_counts to get count of distinct values
         df["Gender"].value_counts()
```

```
Out[98]: M    414259
         F    135809
         Name: Gender, dtype: int64
```

```
In [99]: #sort w.r.t count
         df["Gender"].value_counts(ascending=True)
```

```
Out[99]: F    135809
         M    414259
         Name: Gender, dtype: int64
```

```
In [100…  #lets get age count sorted in ascending order
          df["Age"].value_counts(ascending=False)
```

```
Out[100]: 26-35    219587
          36-45    110013
          18-25     99660
          46-50     45701
          51-55     38501
          55+       21504
          0-17      15102
          Name: Age, dtype: int64
```

```
In [101…  #we can also replace for column values
          df["Gender"]=df["Gender"].replace("F","Female")
          df["Gender"]=df["Gender"].replace("M","Male")
```

```
In [102…  df["Gender"]
```

```
Out[102]: 0         Female
          1         Female
          2         Female
          3         Female
          4           Male
                    ...
          550063      Male
          550064    Female
          550065    Female
          550066    Female
          550067    Female
          Name: Gender, Length: 550068, dtype: object
```

In [103…
```
#verify
df.head()
```

Out[103]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	Female	0-17	10	A	2	
1	1000001	P00248942	Female	0-17	10	A	2	
2	1000001	P00087842	Female	0-17	10	A	2	
3	1000001	P00085442	Female	0-17	10	A	2	
4	1000002	P00285442	Male	55+	16	C	4+	

Task-Generate list of User ID with corresponding age and find the total count of purchases that have done.

In [104…
```
#lets get list first using values anf tolist
df[["User_ID","Age"]].values.tolist()
```

```
Out[104]: [[1000001, '0-17'],
           [1000001, '0-17'],
           [1000001, '0-17'],
           [1000001, '0-17'],
           [1000002, '55+'],
           [1000003, '26-35'],
           [1000004, '46-50'],
           [1000004, '46-50'],
           [1000004, '46-50'],
           [1000005, '26-35'],
           [1000005, '26-35'],
           [1000005, '26-35'],
           [1000005, '26-35'],
           [1000005, '26-35'],
           [1000006, '51-55'],
           [1000006, '51-55'],
           [1000006, '51-55'],
           [1000006, '51-55'],
           [1000007, '36-45'],
           [1000008, '26-35'],
           [1000008, '26-35'],
           [1000008, '26-35'],
           [1000008, '26-35'],
           [1000008, '26-35'],
           [1000008, '26-35'],
           [1000009, '26-35'],
           [1000009, '26-35'],
           [1000009, '26-35'],
           [1000009, '26-35'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000010, '36-45'],
           [1000011, '26-35'],
           [1000011, '26-35'],
           [1000011, '26-35'],
           [1000012, '26-35'],
           [1000012, '26-35'],
           [1000013, '46-50'],
           [1000013, '46-50'],
           [1000013, '46-50'],
           [1000014, '36-45'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000015, '26-35'],
           [1000016, '36-45'],
           [1000016, '36-45'],
           [1000017, '51-55'],
           [1000017, '51-55'],
           [1000017, '51-55'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000018, '18-25'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000019, '0-17'],
           [1000021, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000022, '18-25'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000023, '36-45'],
           [1000024, '26-35'],
           [1000024, '26-35'],
           [1000024, '26-35'],
           [1000025, '18-25'],
           [1000025, '18-25'],
           [1000025, '18-25'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000026, '26-35'],
           [1000027, '26-35'],
           [1000027, '26-35'],
           [1000027, '26-35'],
           [1000027, '26-35'],
           [1000027, '26-35'],
           [1000028, '26-35'],
           [1000028, '26-35'],
           [1000028, '26-35'],
           [1000028, '26-35'],
           [1000028, '26-35'],
           [1000029, '36-45'],
           [1000029, '36-45'],
           [1000029, '36-45'],
           [1000029, '36-45'],
           [1000030, '36-45'],
           [1000030, '36-45'],
           [1000030, '36-45'],
           [1000031, '55+'],
           [1000031, '55+'],
           [1000031, '55+'],
           [1000031, '55+'],
           [1000032, '26-35'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000033, '46-50'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000034, '18-25'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000035, '46-50'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000036, '26-35'],
           [1000037, '26-35'],
           [1000039, '18-25'],
           [1000039, '18-25'],
           [1000041, '18-25'],
           [1000041, '18-25'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000042, '26-35'],
           [1000043, '26-35'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000044, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000045, '46-50'],
           [1000046, '18-25'],
           [1000046, '18-25'],
           [1000046, '18-25'],
           [1000046, '18-25'],
           [1000047, '18-25'],
           [1000047, '18-25'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000048, '26-35'],
           [1000049, '18-25'],
           [1000049, '18-25'],
           [1000049, '18-25'],
           [1000049, '18-25'],
           [1000049, '18-25'],
           [1000049, '18-25'],
           [1000050, '26-35'],
           [1000050, '26-35'],
           [1000051, '0-17'],
           [1000052, '18-25'],
           [1000052, '18-25'],
           [1000052, '18-25'],
           [1000052, '18-25'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000053, '26-35'],
           [1000054, '51-55'],
           [1000054, '51-55'],
           [1000056, '36-45'],
           [1000056, '36-45'],
           [1000056, '36-45'],
           [1000056, '36-45'],
           [1000057, '18-25'],
           [1000057, '18-25'],
           [1000057, '18-25'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000058, '26-35'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000059, '51-55'],
           [1000060, '51-55'],
           [1000060, '51-55'],
           [1000061, '26-35'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000062, '36-45'],
           [1000063, '18-25'],
           [1000063, '18-25'],
           [1000063, '18-25'],
           [1000063, '18-25'],
           [1000064, '18-25'],
           [1000065, '36-45'],
           [1000065, '36-45'],
           [1000065, '36-45'],
           [1000065, '36-45'],
           [1000065, '36-45'],
           [1000065, '36-45'],
           [1000066, '26-35'],
           [1000067, '51-55'],
           [1000067, '51-55'],
           [1000067, '51-55'],
           [1000068, '18-25'],
           [1000068, '18-25'],
           [1000068, '18-25'],
           [1000069, '26-35'],
           [1000069, '26-35'],
           [1000070, '18-25'],
           [1000070, '18-25'],
           [1000071, '26-35'],
           [1000071, '26-35'],
           [1000071, '26-35'],
           [1000071, '26-35'],
           [1000072, '46-50'],
           [1000072, '46-50'],
           [1000073, '18-25'],
           [1000073, '18-25'],
           [1000073, '18-25'],
           [1000073, '18-25'],
           [1000073, '18-25'],
           [1000074, '36-45'],
           [1000074, '36-45'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000075, '0-17'],
           [1000076, '36-45'],
           [1000076, '36-45'],
           [1000076, '36-45'],
           [1000076, '36-45'],
           [1000076, '36-45'],
           [1000077, '18-25'],
           [1000077, '18-25'],
           [1000077, '18-25'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000078, '46-50'],
           [1000079, '46-50'],
           [1000079, '46-50'],
           [1000080, '55+'],
           [1000080, '55+'],
           [1000080, '55+'],
           [1000081, '26-35'],
           [1000082, '26-35'],
           [1000082, '26-35'],
           [1000083, '26-35'],
           [1000083, '26-35'],
           [1000083, '26-35'],
           [1000083, '26-35'],
           [1000083, '26-35'],
           [1000084, '18-25'],
           [1000084, '18-25'],
           [1000084, '18-25'],
           [1000084, '18-25'],
           [1000085, '18-25'],
           [1000086, '0-17'],
           [1000086, '0-17'],
           [1000087, '26-35'],
           [1000087, '26-35'],
           [1000087, '26-35'],
           [1000088, '46-50'],
           [1000088, '46-50'],
           [1000088, '46-50'],
           [1000089, '55+'],
           [1000089, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000090, '55+'],
           [1000091, '36-45'],
           [1000091, '36-45'],
           [1000091, '36-45'],
           [1000091, '36-45'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000092, '18-25'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000093, '26-35'],
           [1000094, '26-35'],
           [1000095, '46-50'],
           [1000095, '46-50'],
           [1000096, '26-35'],
           [1000096, '26-35'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000097, '36-45'],
           [1000099, '0-17'],
           [1000099, '0-17'],
           [1000099, '0-17'],
           [1000099, '0-17'],
           [1000099, '0-17'],
           [1000100, '36-45'],
           [1000100, '36-45'],
           [1000100, '36-45'],
           [1000101, '18-25'],
           [1000101, '18-25'],
           [1000101, '18-25'],
           [1000101, '18-25'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000102, '36-45'],
           [1000103, '46-50'],
           [1000103, '46-50'],
           [1000103, '46-50'],
           [1000103, '46-50'],
           [1000105, '46-50'],
           [1000105, '46-50'],
           [1000105, '46-50'],
           [1000105, '46-50'],
           [1000106, '36-45'],
           [1000106, '36-45'],
           [1000106, '36-45'],
           [1000107, '46-50'],
           [1000107, '46-50'],
           [1000107, '46-50'],
           [1000107, '46-50'],
           [1000107, '46-50'],
           [1000107, '46-50'],
           [1000108, '26-35'],
           [1000109, '46-50'],
           [1000109, '46-50'],
           [1000109, '46-50'],
           [1000110, '26-35'],
           [1000111, '36-45'],
           [1000111, '36-45'],
           [1000112, '26-35'],
           [1000113, '18-25'],
           [1000113, '18-25'],
           [1000114, '26-35'],
           [1000114, '26-35'],
           [1000116, '26-35'],
           [1000116, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000117, '26-35'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000118, '36-45'],
           [1000119, '0-17'],
           [1000119, '0-17'],
           [1000119, '0-17'],
           [1000119, '0-17'],
           [1000119, '0-17'],
           [1000120, '26-35'],
           [1000121, '36-45'],
           [1000121, '36-45'],
           [1000122, '18-25'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000123, '36-45'],
           [1000124, '55+'],
           [1000125, '46-50'],
           [1000125, '46-50'],
           [1000125, '46-50'],
           [1000125, '46-50'],
           [1000125, '46-50'],
           [1000125, '46-50'],
           [1000126, '18-25'],
           [1000126, '18-25'],
           [1000127, '46-50'],
           [1000127, '46-50'],
           [1000127, '46-50'],
           [1000127, '46-50'],
           [1000127, '46-50'],
           [1000128, '55+'],
           [1000129, '26-35'],
           [1000129, '26-35'],
           [1000129, '26-35'],
           [1000129, '26-35'],
           [1000129, '26-35'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000130, '36-45'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000131, '18-25'],
           [1000132, '26-35'],
           [1000132, '26-35'],
           [1000132, '26-35'],
           [1000132, '26-35'],
           [1000132, '26-35'],
           [1000133, '26-35'],
           [1000133, '26-35'],
           [1000133, '26-35'],
           [1000133, '26-35'],
           [1000133, '26-35'],
           [1000134, '26-35'],
           [1000134, '26-35'],
           [1000134, '26-35'],
           [1000134, '26-35'],
           [1000135, '18-25'],
           [1000135, '18-25'],
           [1000135, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000136, '18-25'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000137, '46-50'],
           [1000138, '18-25'],
           [1000138, '18-25'],
           [1000138, '18-25'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000139, '26-35'],
           [1000140, '36-45'],
           [1000140, '36-45'],
           [1000142, '26-35'],
           [1000142, '26-35'],
           [1000142, '26-35'],
           [1000142, '26-35'],
           [1000142, '26-35'],
           [1000143, '18-25'],
           [1000143, '18-25'],
           [1000143, '18-25'],
           [1000145, '18-25'],
           [1000145, '18-25'],
           [1000145, '18-25'],
           [1000145, '18-25'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000146, '36-45'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000147, '18-25'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000148, '51-55'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000149, '26-35'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000150, '36-45'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000151, '26-35'],
           [1000152, '18-25'],
           [1000153, '0-17'],
           [1000153, '0-17'],
           [1000153, '0-17'],
           [1000154, '51-55'],
           [1000154, '51-55'],
           [1000154, '51-55'],
           [1000154, '51-55'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000155, '36-45'],
           [1000156, '46-50'],
           [1000156, '46-50'],
           [1000156, '46-50'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000157, '36-45'],
           [1000158, '55+'],
           [1000158, '55+'],
           [1000159, '46-50'],
           [1000160, '36-45'],
           [1000160, '36-45'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000161, '46-50'],
           [1000162, '18-25'],
           [1000162, '18-25'],
           [1000162, '18-25'],
           [1000162, '18-25'],
           [1000162, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000163, '18-25'],
           [1000165, '18-25'],
           [1000165, '18-25'],
           [1000165, '18-25'],
           [1000165, '18-25'],
           [1000165, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000166, '18-25'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000169, '26-35'],
           [1000170, '26-35'],
           [1000170, '26-35'],
           [1000171, '51-55'],
           [1000172, '26-35'],
           [1000173, '26-35'],
           ...]
```

In [105…

```
#lets check for count of purchase for all distint User_ID and age combination.
df[["User_ID","Age"]].value_counts()
```

Out[105]:

```
User_ID  Age
1001680  26-35    1026
1004277  36-45     979
1001941  36-45     898
1001181  36-45     862
1000889  46-50     823
                  ...
1002111  55+         7
1005391  26-35       7
1002690  26-35       7
1005608  18-25       7
1000708  26-35       6
Length: 5891, dtype: int64
```

Aggregation in pandas

Task-Get different statistical values for Purchase column

In [106…
```
import numpy as np
```

In [107…
```
df["Purchase"].describe()
```

Out[107]:
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

Task-Find the total amount generated via website by selling product

we can apply aggregation on a single column of a dataframe

In [108…
```
#lets use np.sum aggregation to get total purchased amount
df["Purchase"].aggregate(np.sum)
```

Out[108]:
```
5095812742
```

In [109…
```
#we can also apply multiple function on a single column of a dataframe
#find sum and mean value after doing aggregation over purchase column
df["Purchase"].aggregate([np.sum,np.mean])
```

Out[109]:
```
sum     5.095813e+09
mean    9.263969e+03
Name: Purchase, dtype: float64
```

In [110…
```
#we can also apply aggregation on multiple columns of a dataframe.

#find mean for "Product_Category_1","Product_Category_2","Product_Category_3"

df[["Product_Category_1","Product_Category_2","Product_Category_3"]].aggregate(np.r
```

Out[110]:
```
Product_Category_1     5.404270
Product_Category_2     9.863190
Product_Category_3    12.650723
dtype: float64
```

In [111…
```
#we can also apply multiole function on multiple columns of a dataframe.
#find mean and sum for "Product_Category_1","Product_Category_2","Product_Category_

df[["Product_Category_1","Product_Category_2","Product_Category_3"]].aggregate([np.
```

Out[111]:

	Product_Category_1	Product_Category_2	Product_Category_3
sum	2.972716e+06	5.425425e+06	6.958758e+06
mean	5.404270e+00	9.863190e+00	1.265072e+01

Function application in Pandas

Task-Tag records to "High focused" transaction where purchase amount has been more than 5000.Remaining can be tagged as general transaction.

-Row or column wise function application apply()

In [112…

```
#use apply function on Product_Category_1
df.Product_Category_1.apply(lambda x:x*10)
```

Out[112]:

```
0          30
1          10
2         120
3         120
4          80
         ...
550063    200
550064    200
550065    200
550066    200
550067    200
Name: Product_Category_1, Length: 550068, dtype: int64
```

In [113…

```
#verify
df.head()
```

Out[113]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	Female	0-17	10	A	2	
1	1000001	P00248942	Female	0-17	10	A	2	
2	1000001	P00087842	Female	0-17	10	A	2	
3	1000001	P00085442	Female	0-17	10	A	2	
4	1000002	P00285442	Male	55+	16	C	4+	

In [114…

```
#lets add new column as "Category" which will have tags based on purchase amount
df["Category"]=df.Purchase.apply(lambda x :"High Focused" if x>5000 else "General")
```

In [115…

```
df.head()
```

Out[115]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	Female	0-17	10	A	2	
1	1000001	P00248942	Female	0-17	10	A	2	
2	1000001	P00087842	Female	0-17	10	A	2	
3	1000001	P00085442	Female	0-17	10	A	2	
4	1000002	P00285442	Male	55+	16	C	4+	

In [116…

```
#lets check value for highly focused row
df.Category.value_counts()
```

Out[116]:
```
High Focused    455145
General          94923
Name: Category, dtype: int64
```

Pandas GroupBy operations

Task-Based on gender,check the total purchased amount and average purchasing amount

In [117…
```
#use groupby on top of gender column
df.groupby("Gender")
```

Out[117]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002E56DB35EB0>
```

In [118…
```
df.groupby("Gender").groups
```

Out[118]:
```
{'Female': [0, 1, 2, 3, 14, 15, 16, 17, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3
9, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 65, 66, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 124, 125, 126, 147, 148, 149, 150, 151, 156, 157, 158,
163, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 219, 222, 223, 248, 249, 25
0, 251, 252, 253, 254, 255, 256, 257, 297, 298, 299, 355, 356, 357, 358, 359, 360,
361, 362, 363, 364, 365, 366, 367, 368, 369, 373, ...], 'Male': [4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 50, 51, 52, 53, 54, 5
5, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 127, 128, 129, 13
0, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
152, 153, ...]}
```

In [119…
```
#Group by with multiple columns

#get groups based on gender and age combination

df.groupby(["Gender","Age"]).groups
```

```
Out[119]: {('Female', '0-17'): [0, 1, 2, 3, 299, 423, 424, 425, 426, 427, 428, 429, 430, 43
          1, 432, 467, 468, 539, 540, 541, 542, 543, 617, 618, 619, 620, 621, 1150, 1151, 13
          04, 1305, 1306, 2905, 2907, 3010, 3715, 3804, 3805, 3806, 3807, 3808, 3835, 3836,
          4551, 4552, 4553, 4554, 4555, 5453, 6431, 6759, 6760, 6761, 6762, 6763, 6764, 676
          5, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775, 6776, 6777, 6778, 6
          779, 6780, 6781, 6782, 6783, 6784, 6785, 6786, 6787, 6788, 6789, 6790, 6791, 6792,
          6793, 6794, 6795, 6796, 6797, 6798, 6799, 6800, 6801, 6802, 6803, 6804, 6805, 680
          6, 6807, 6808, ...], ('Female', '18-25'): [70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
          80, 81, 82, 83, 84, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 222, 223, 49
          5, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 547, 548,
          549, 550, 625, 910, 911, 912, 913, 914, 1046, 1228, 1267, 1268, 1269, 1490, 1491,
          1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1552, 1553, 1554, 1555, 155
          6, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1
          678, 1822, 1903, 1904, 1905, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1959,
          ...], ('Female', '26-35'): [47, 48, 49, 124, 125, 126, 147, 148, 149, 150, 151, 16
          3, 219, 297, 298, 406, 407, 454, 457, 458, 459, 460, 461, 529, 530, 585, 586, 691,
          692, 693, 694, 695, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 74
          1, 742, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854,
          855, 856, 857, 858, 859, 860, 861, 862, 863, 1033, 1034, 1035, 1036, 1037, 1038, 1
          039, 1040, 1041, 1042, 1043, 1044, 1045, 1085, 1086, 1087, 1088, 1364, 1365, 1369,
          1565, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, ...], ('Female', '36-4
          5'): [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 65,
          66, 156, 157, 158, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 38
          5, 386, 387, 531, 532, 533, 534, 535, 536, 537, 538, 566, 567, 568, 743, 744, 757,
          758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 77
          4, 775, 776, 777, 778, 779, 780, 1187, 1188, 1189, 1190, 1191, 1229, 1230, 1231, 1
          232, 1233, 1652, 1653, 1741, 1770, 1771, 1772, 1773, 1774, 2197, 2198, 2199, 2200,
          2201, 2202, 2203, ...], ('Female', '46-50'): [248, 249, 250, 251, 252, 253, 254, 2
          55, 256, 257, 414, 415, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 472, 47
          3, 474, 654, 655, 656, 657, 658, 717, 718, 719, 720, 721, 722, 723, 724, 725, 879,
          880, 881, 895, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1095, 1096, 1097, 1098, 1
          099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112,
          1113, 1114, 1115, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1434, 143
          5, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2
          562, 2563, 2564, 2565, 2566, ...], ('Female', '51-55'): [14, 15, 16, 17, 355, 356,
          357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 400, 401, 402, 99
          7, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1
          480, 1957, 1958, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
          1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2026, 202
          7, 2028, 2782, 2783, 2784, 3526, 3527, 3528, 3650, 3651, 3652, 3653, 3654, 3993, 3
          994, 3995, 3996, 4177, 4178, 4179, 4180, 4755, 4901, 4902, 4903, 5625, 5626, 5630,
          5631, 5632, 5633, 5884, 5885, 5886, 5887, 5888, 5889, ...], ('Female', '55+'): [47
          5, 476, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 19
          73, 1974, 1975, 1976, 1977, 1981, 1982, 2139, 2140, 2141, 2142, 2227, 2228, 2229,
          2230, 2231, 2232, 3123, 3124, 3125, 3126, 3127, 3128, 3129, 3130, 3131, 3132, 313
          3, 3134, 3655, 3656, 3657, 3658, 3659, 3660, 3687, 3688, 3689, 3690, 4693, 5444, 5
          445, 5446, 5447, 5448, 5449, 5450, 5451, 5452, 5594, 5595, 5596, 5597, 5732, 5733,
          5734, 5735, 5736, 5737, 5738, 5739, 5740, 5741, 5742, 5743, 5744, 5745, 5765, 576
          6, 5767, 5768, 5769, 6077, 6078, 6404, 6405, 6406, 6407, 6408, 6409, 6931, 7780, 7
          781, 7782, 7783, 8223, ...], ('Male', '0-17'): [85, 86, 87, 88, 89, 90, 91, 92, 9
          3, 94, 95, 96, 865, 866, 867, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 224
          4, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 3014, 3015, 3016, 3017, 3018, 3568, 3
          569, 3570, 3571, 4375, 4526, 4527, 4528, 4529, 4530, 4531, 4532, 4647, 4648, 4649,
          4650, 4651, 4652, 4653, 4654, 4655, 4656, 4698, 4699, 4771, 5059, 5060, 5061, 506
          2, 5063, 5064, 5065, 5066, 5352, 5361, 5362, 5435, 5436, 5437, 5438, 5439, 5440, 5
          441, 5624, 5725, 5821, 5822, 5823, 5824, 5919, 5920, 5921, 5922, 5923, 5924, 5925,
          6032, 6033, 6112, 6113, 6114, 6115, 6520, 6521, ...], ('Male', '18-25'): [97, 98,
          99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 127, 128, 129, 22
          0, 221, 258, 259, 260, 261, 262, 263, 291, 292, 293, 294, 295, 296, 300, 301, 302,
          303, 339, 340, 341, 388, 389, 390, 391, 392, 403, 404, 405, 408, 409, 416, 417, 41
          8, 419, 420, 438, 439, 440, 462, 463, 464, 465, 466, 583, 584, 652, 653, 676, 677,
          678, 679, 680, 681, 682, 683, 684, 685, 700, 701, 702, 703, 704, 705, 706, 707, 70
          8, 709, 710, 711, 712, 713, 714, 715, 716, 726, 727, 728, 750, 751, 752, 753,
          ...], ('Male', '26-35'): [5, 9, 10, 11, 12, 13, 19, 20, 21, 22, 23, 24, 25, 26, 2
          7, 28, 50, 51, 56, 57, 58, 59, 60, 61, 62, 63, 64, 130, 131, 132, 133, 134, 135, 1
          36, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 196, 197, 198, 199, 200, 20
          1, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217,
          218, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 264, 265, 266, 267, 26
          8, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284,
          285, ...], ('Male', '36-45'): [18, 55, 112, 113, 114, 115, 116, 117, 118, 119, 12
          0, 121, 122, 123, 152, 153, 154, 155, 335, 336, 337, 338, 393, 394, 395, 396, 397,
          398, 421, 422, 433, 434, 435, 436, 437, 491, 492, 493, 494, 544, 545, 546, 551, 55
          2, 553, 554, 555, 556, 557, 580, 581, 605, 606, 607, 608, 609, 610, 611, 612, 613,
          614, 615, 616, 623, 624, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 63
          7, 638, 639, 640, 641, 642, 643, 644, 665, 666, 667, 668, 669, 670, 671, 672, 673,
          674, 675, 830, 831, 832, 833, 834, ...], ('Male', '46-50'): [6, 7, 8, 52, 53, 54,
          164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 189, 19
          0, 191, 192, 193, 194, 195, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
          246, 247, 527, 528, 558, 559, 560, 561, 562, 563, 564, 565, 569, 570, 571, 572, 57
          3, 574, 576, 577, 578, 646, 647, 648, 649, 650, 651, 898, 899, 900, 901, 902, 903,
          904, 905, 906, 907, 908, 909, 1057, 1058, 1089, 1090, 1091, 1307, 1308, 1309, 132
          3, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336,
          ...], ('Male', '51-55'): [67, 68, 69, 333, 334, 370, 371, 788, 789, 790, 791, 792,
          793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 868, 86
          9, 870, 871, 1047, 1048, 1049, 1050, 1486, 1487, 1488, 1489, 1503, 1504, 1505, 150
          6, 1681, 1682, 1683, 1738, 1739, 1740, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1
          782, 1815, 1816, 1817, 1818, 1819, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
          2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 204
          2, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2
          056, 2124, 2175, ...], ('Male', '55+'): [4, 159, 160, 161, 162, 451, 452, 453, 47
          7, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 645, 659, 893,
          894, 1051, 1052, 1053, 1054, 1116, 1117, 1118, 1119, 1559, 1560, 1792, 1793, 1794,
          1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1978, 197
          9, 1980, 2016, 2017, 2018, 2096, 2097, 2322, 2510, 2614, 2615, 2766, 2767, 2768, 2
          769, 2770, 2771, 2793, 2794, 2795, 2796, 2797, 2798, 2799, 3011, 3478, 3837, 3838,
          3839, 3840, 3841, 3842, 3843, 3844, 3845, 4175, 4176, 4423, 4424, 4425, 4426, 469
          4, 4695, 4696, 5052, 5053, 5083, 5084, ...]}
```

In [120…

```
#apply aggregation function sum with groupby
df.groupby("Gender").sum()
```

Out[120]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Produ
Gender						
Female	136234060927	915426	56988	776517	1356094.0	
Male	415500008355	3527312	168349	2196199	4069331.0	

In [121…

```
#use np.sum
df.groupby("Gender").agg(np.sum)
```

Out[121]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Produ
Gender						
Female	136234060927	915426	56988	776517	1356094.0	
Male	415500008355	3527312	168349	2196199	4069331.0	

In [122…

```
#get total purchased amount
df.groupby("Gender")["Purchase"].agg(np.sum)
```

Out[122]:
```
Gender
Female    1186232642
Male      3909580100
Name: Purchase, dtype: int64
```

In [123…
```
#get sum as well as mean
df.groupby("Gender")["Purchase"].agg([np.sum,np.mean])
```

Out[123]:

	sum	mean
Gender		
Female	1186232642	8734.565765
Male	3909580100	9437.526040

In [124…
```
#we can also apply function on top of group
df[df.groupby("Gender")["Purchase"].apply(lambda x: x>10000)]
```

Out[124]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
1	1000001	P00248942	Female	0-17	10	A	2
5	1000003	P00193542	Male	26-35	15	A	3
6	1000004	P00184942	Male	46-50	7	B	2
7	1000004	P00346142	Male	46-50	7	B	2
8	1000004	P0097242	Male	46-50	7	B	2
...	...	...	...	...	...	...	...
545892	1006037	P00148642	Female	46-50	1	C	4+
545896	1006037	P00183142	Female	46-50	1	C	4+
545904	1006040	P00081142	Male	26-35	6	B	2
545908	1006040	P00127642	Male	26-35	6	B	2
545914	1006040	P00217442	Male	26-35	6	B	2

189450 rows × 13 columns

Task-Create new columns based on City_Category values and drop the original column

Handling multi-Valued categorical columns

In [125…
```
#Check different values for city category
df.City_Category.value_counts()
```

Out[125]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [126…
```
#apply get_dummies function to get new columns
dummy_df=pd.get_dummies(df.City_Category,drop_first=True)
```

In [127…
```
dummy_df.head()
```

Out[127]:

	B	C
0	0	0
1	0	0
2	0	0
3	0	0
4	0	1

In [128…
```
#concatenate both dataframes
df=pd.concat([df,dummy_df],axis=1)
```

In [129…
```
#verify
df
```

Out[129]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	Female	0-17	10	A	2
1	1000001	P00248942	Female	0-17	10	A	2
2	1000001	P00087842	Female	0-17	10	A	2
3	1000001	P00085442	Female	0-17	10	A	2
4	1000002	P00285442	Male	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	Male	51-55	13	B	1
550064	1006035	P00375436	Female	26-35	1	C	3
550065	1006036	P00375436	Female	26-35	15	B	4+
550066	1006038	P00375436	Female	55+	1	C	2
550067	1006039	P00371644	Female	46-50	0	B	4+

550068 rows × 15 columns

In [130…

```
#drop original one
df.drop(["City_Category"],axis=1,inplace=True)
```

In [131…

```
#verify
df.head()
```

Out[131]:

	User_ID	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Proc
0	1000001	P00069042	Female	0-17	10	2	0	
1	1000001	P00248942	Female	0-17	10	2	0	
2	1000001	P00087842	Female	0-17	10	2	0	
3	1000001	P00085442	Female	0-17	10	2	0	
4	1000002	P00285442	Male	55+	16	4+	0	

In [132…

```
#also verify shape
df.shape
```

Out[132]:

```
(550068, 14)
```

-Generated descriptive statistical values